Software Design and Architecture LAB

SDA Lab Manual # 11

Border Layout and Grid Layout in Java GUI

Instructor: Fariba Laiq

Fast National University of Computer and Emerging Sciences, Peshawar
Department of Computer Science & Software Engineering

Border Layout

The class **BorderLayout** arranges the components to fit in the five regions: east, west, north, south, and center. Each region can contain only one component and each component in each region is identified by the corresponding constant NORTH, SOUTH, EAST, WEST, and CENTER.

Class Constructors

Sr.No.	Constructor & Description
1	**BorderLayout()** Constructs a new border layout with no gaps between the components.
2	**BorderLayout(int hgap, int vgap)** Constructs a border layout with the specified gaps between the components.

This code is a simple Java program that creates a GUI (graphical user interface) window using the Swing library.

The MyGUI class defines a constructor which calls the prepareGUI method. The prepareGUI method creates a JFrame object, which is the main window of the GUI. It then gets the content pane of the JFrame using getContentPane() method and sets a BorderLayout on it using setLayout() method.

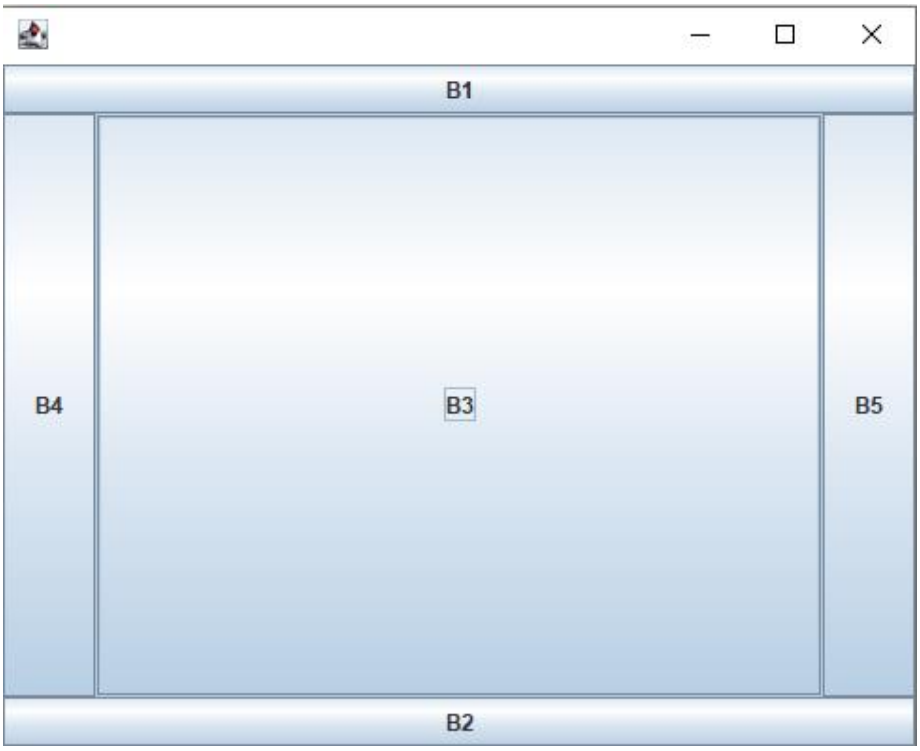
The BorderLayout divides the content pane into five regions: North, South, Center, West, and East. The prepareGUI method then creates five JButton objects, labeled B1 to B5. These buttons are added to the content pane using the add() method with the appropriate BorderLayout constraints.

MyGUI.java

```
import javax.swing.*;
import java.awt.*;

public class MyGUI {
    MyGUI()
    {
        prepareGUI();
    }

    private void prepareGUI() {
        JFrame jframe = new JFrame();
        Container c = jframe.getContentPane();
        BorderLayout borderLayout= new BorderLayout();
        c.setLayout(borderLayout);

        JButton b1=new JButton("B1");
        JButton b2=new JButton("B2");
        JButton b3=new JButton("B3");
        JButton b4=new JButton("B4");
        JButton b5=new JButton("B5");

        c.add(b1, BorderLayout.NORTH);
        c.add(b2, BorderLayout.SOUTH);
        c.add(b3, BorderLayout.CENTER);
        c.add(b4, BorderLayout.WEST);
        c.add(b5, BorderLayout.EAST);

        jframe.setVisible(true);
        jframe.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        jframe.setSize(500,400);
    }

}
```

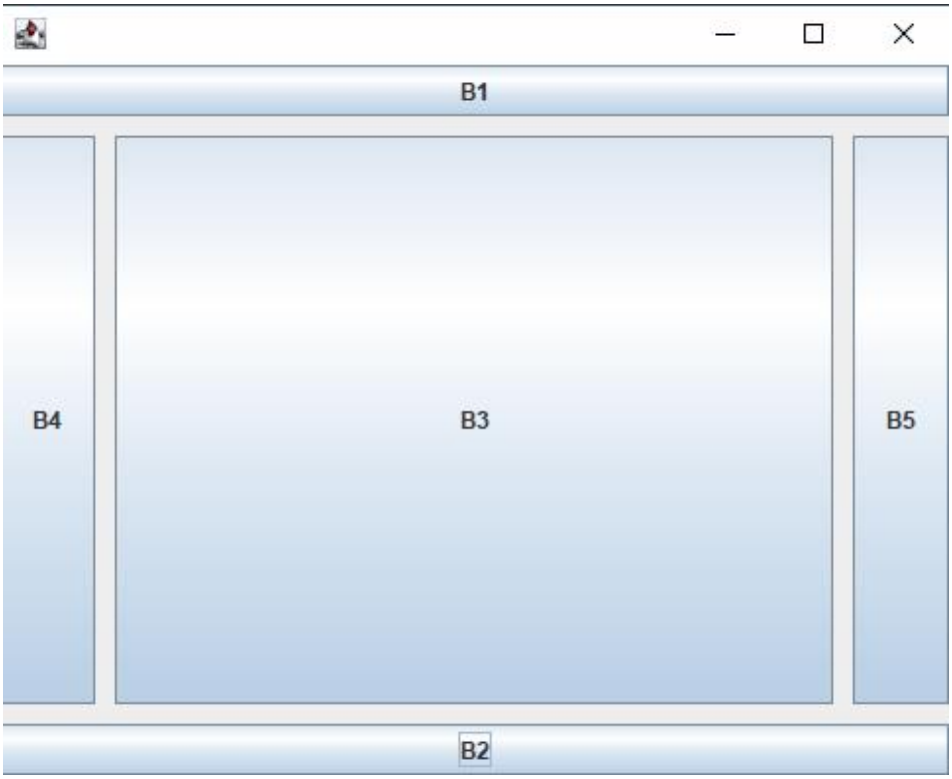
Main.java

```
public class Main {
    public static void main(String[] args) {
        MyGUI g = new MyGUI();
    }
}
```



If you want to set some gap between the components, use the following method.

```
borderLayout.setHgap(10);
borderLayout.setVgap(10);
```



Grid Layout

The Java GridLayout class is used to arrange the components in a rectangular grid i-e in the form of rows and columns.

Constructors of GridLayout class

1. **GridLayout():** creates a grid layout with one column per component in a row.
2. **GridLayout(int rows, int columns):** creates a grid layout with the given rows and columns but no gaps between the components.
3. **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

Example of GridLayout class: Using GridLayout() Constructor

The GridLayout() constructor creates only one row. The following example shows the usage of the parameterless constructor.

MyGUI.java

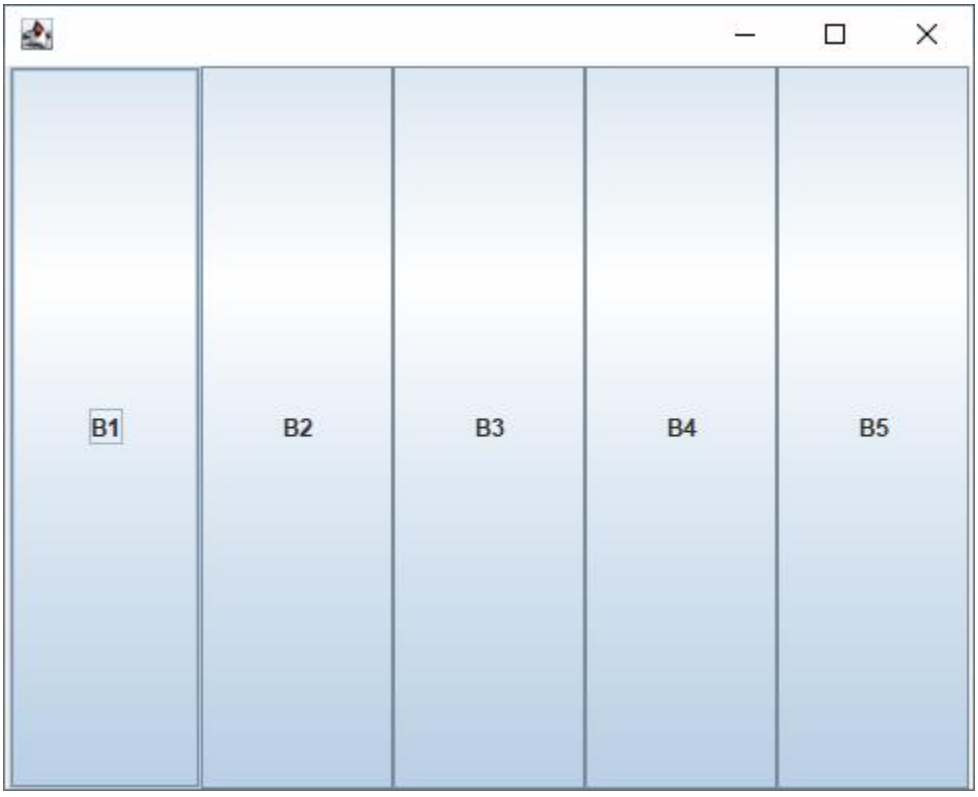
```
import javax.swing.*;
import java.awt.*;

public class MyGUI {
    MyGUI()
    {
        prepareGUI();
    }

    private void prepareGUI() {
        JFrame jframe = new JFrame();
        Container c = jframe.getContentPane();
        GridLayout g=new GridLayout();
        c.setLayout(g);

        JButton b1=new JButton("B1");
        JButton b2=new JButton("B2");
        JButton b3=new JButton("B3");
        JButton b4=new JButton("B4");
        JButton b5=new JButton("B5");

        c.add(b1);
        c.add(b2);
        c.add(b3);
        c.add(b4);
        c.add(b5);

        jframe.setVisible(true);
```

```
        jframe.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        jframe.setSize(500,400);
    }

}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        MyGUI g = new MyGUI();
    }
}
```

Output:



Example of GridLayout class: Using GridLayout(int rows, int columns) Constructor

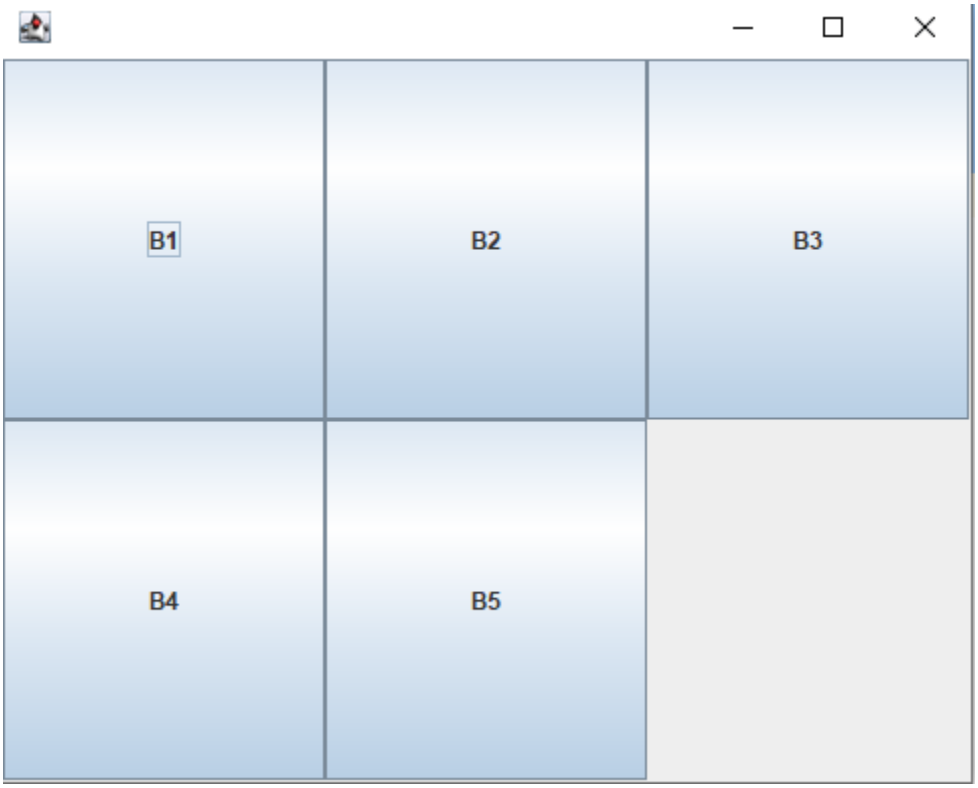
MyGUI.java

```
import javax.swing.*;
import java.awt.*;

public class MyGUI {
    MyGUI()
    {
        prepareGUI();
    }

    private void prepareGUI() {
        JFrame jframe = new JFrame();
        Container c = jframe.getContentPane();
        GridLayout g=new GridLayout(2,3);
        c.setLayout(g);

        JButton b1=new JButton("B1");
        JButton b2=new JButton("B2");
        JButton b3=new JButton("B3");
        JButton b4=new JButton("B4");
        JButton b5=new JButton("B5");

        c.add(b1);
        c.add(b2);
        c.add(b3);
        c.add(b4);
        c.add(b5);

        jframe.setVisible(true);
        jframe.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        jframe.setSize(500,400);
    }

}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        MyGUI g = new MyGUI();
    }
}
```

Output:



Example of GridLayout class: Using GridLayout(int rows, int columns, int hgap, int vgap) Constructor

MyGUI.java

```
import javax.swing.*;
import java.awt.*;

public class MyGUI {
    MyGUI()
    {
        prepareGUI();
    }

    private void prepareGUI() {
        JFrame jframe = new JFrame();
        Container c = jframe.getContentPane();
        GridLayout g=new GridLayout(2,3, 20, 20);
        c.setLayout(g);
```

```
        JButton b1=new JButton("B1");
        JButton b2=new JButton("B2");
        JButton b3=new JButton("B3");
        JButton b4=new JButton("B4");
        JButton b5=new JButton("B5");

        c.add(b1);
        c.add(b2);
        c.add(b3);
        c.add(b4);
        c.add(b5);

        jframe.setVisible(true);
        jframe.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        jframe.setSize(500,400);
    }

}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        MyGUI g = new MyGUI();
    }
}
```

Creating the UI of a Simple Calculator using Border Layout + Grid Layout

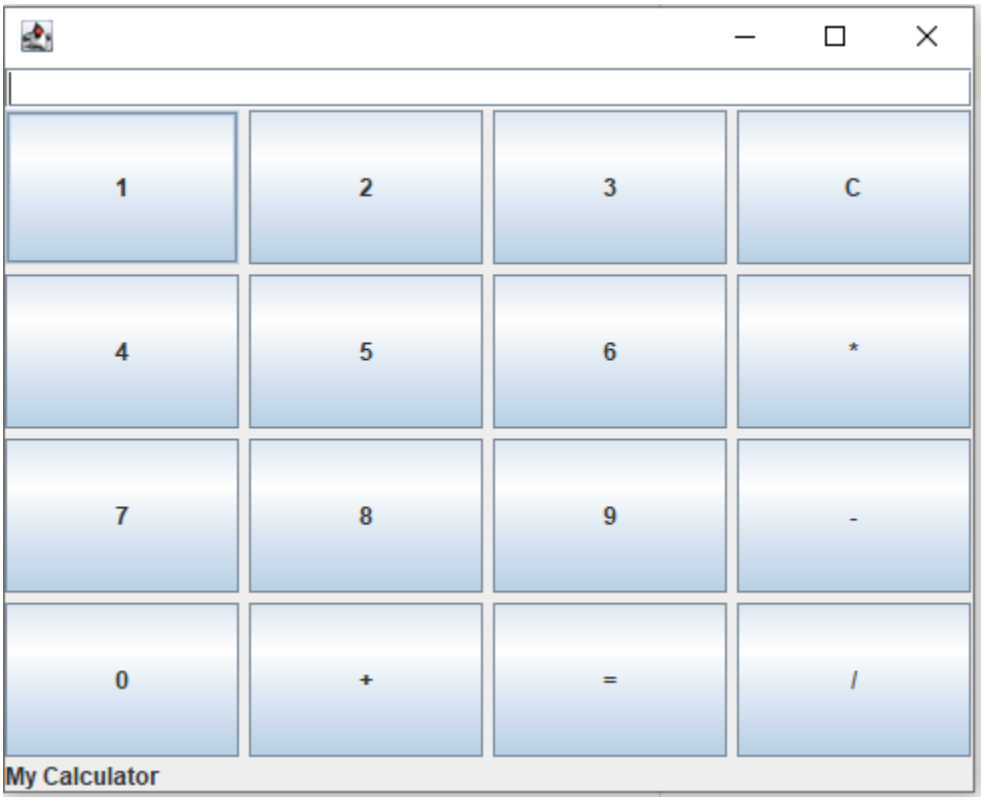
The code starts with the MyGUI class constructor, which calls the prepareGUI() method to set up the GUI. The prepareGUI() method creates a new JFrame object, which is the top-level container for the GUI. The getContentPane() method is called to get the content pane of the JFrame, and a new BorderLayout object is created and set as the layout manager for the content pane.

A new JPanel object is created to hold the buttons for the calculator, as we are creating a nested layout. Inside the border layout center we have another grid layout created with 4 rows and 4 columns, and 5 pixels of horizontal and vertical spacing between the buttons. The JPanel is given this GridLayout object as its layout manager using the setLayout() method. The buttons are added to the grid layout in a row wise fashion.

The JLabel object is created with the text "My Calculator", and a new JTextField object is created to display the input and output of the calculator. The JPanel object is added to the center of the content pane using the add() method with the BorderLayout.CENTER parameter, and the JLabel and JTextField objects are added to the south and north of the content pane, respectively, using the BorderLayout.SOUTH and BorderLayout.NORTH parameters.

The code then creates 16 JButton objects, with labels for the numbers 0-9, and for the arithmetic operators +, -, *, and /, as well as a "C" button to clear the input field, and an "=" button to calculate the result. The JButton objects are added to the JPanel using the add() method in the order that they should appear on the calculator.

Finally, the setVisible() method is called to make the JFrame visible, and the setDefaultCloseOperation() method is called with the WindowConstants.EXIT_ON_CLOSE parameter to close the application when the user closes the window. The setSize() method is called to set the size of the JFrame to 500 pixels wide by 400 pixels tall.



Inserting an Image into JFrame

There are a lot of ways to insert an image. Below is one of them.

Using ImageIcon and JLabel:

The following code reads an image file named "profile.png" using the ImageIO.read() method and scales it to a size of 100x100 pixels using the Image.getScaledInstance() method with Image.SCALE_SMOOTH flag. In particular, Image.SCALE_SMOOTH indicates that the image should be scaled using a high-quality image scaling algorithm that produces a smooth, good-quality result. This algorithm may take longer to compute and use more memory than other scaling algorithms, but it results in a better-looking image.

Then we create an ImageIcon with the scaled image and sets it as the icon of a JLabel using the setIcon() method, and sets the preferred size of the JLabel to 550 pixels width and 100 pixels height, and horizontally centers it using setHorizontalAlignment() method.

MyGUI.java

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.IOException;

public class MyGUI extends JFrame {
    MyGUI() throws IOException {
        prepareGUI();
    }

    private void prepareGUI() throws IOException {
        Container c = getContentPane();
        c.setLayout((new FlowLayout(FlowLayout.CENTER, 0, 10)));

        Image image = ImageIO.read(new File("profile.png"));

        // Scale the image to fit within the label
        Image scaledImage = image.getScaledInstance(100, 100,
Image.SCALE_SMOOTH);

        // Create an ImageIcon with the scaled image
        ImageIcon icon = new ImageIcon(scaledImage);

        // Set the ImageIcon as the icon of the label
        JLabel l = new JLabel();
        l.setIcon(icon);
        l.setPreferredSize(new Dimension(550, 100));
        l.setHorizontalAlignment(SwingConstants.CENTER);

        JLabel uname = new JLabel("Username");
        JLabel pass = new JLabel("Password");

        JTextField username = new JTextField();
        username.setColumns(50);

        JPasswordField password = new JPasswordField();
        password.setColumns(50);

        JButton login = new JButton("Login");

        c.add(l);
        c.add(uname);
        c.add(username);
        c.add(pass);
        c.add(password);
        c.add(login);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(600, 350);
```

```
    }

}
```

Main.java

```
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        MyGUI g = new MyGUI();
    }
}
```

Output: